Good NEWS: Partitioning a Simple Polygon by Compass Directions

Marc van Kreveld

Iris Reinbacher

institute of information and computing sciences, utrecht university

technical report UU-CS-2003-020

www.cs.uu.nl

Good NEWS:
Partitioning a Simple Polygon by Compass Directions*

Marc van Kreveld
marc@cs.uu.nl

Iris Reinbacher
iris@cs.uu.nl

Abstract

Motivated by geographic information retrieval, we study the problem of partitioning a simple polygon into four parts that can be considered as the North, East, West, and South. We list criteria for such partitionings, propose formalizations into geometric problems, and give efficient algorithms. An implementation and tests on country outlines show the results for three different partitionings.

1 Introduction

Nearly all Internet search engines use some form of term matching to create a list of hits and rank the query results. In many cases this yields very good results. However, when a user asks a question like: "What ruins can be found in Eastern Greece?" or: "Which castles are in the proximity of Paris?", standard search engines are less appropriate. The problem is that concepts like "Eastern Greece" and "proximity of Paris" use a spatial relationship with respect to a geographical object, and the query terms "Eastern" and "proximity" themselves are not relevant for term matching.

Problems like the one indicated above have led to research on geographic information retrieval [6, 14, 15, 20]. An effort to address spatial searches on the Internet and to build a spatially-aware search engine will be made in the SPIRIT project [13]. By using geometric footprints associated to Web pages, geographic ontologies, methods to determine spatial relationships, and query expansion, spatial searches can most likely be performed much better.

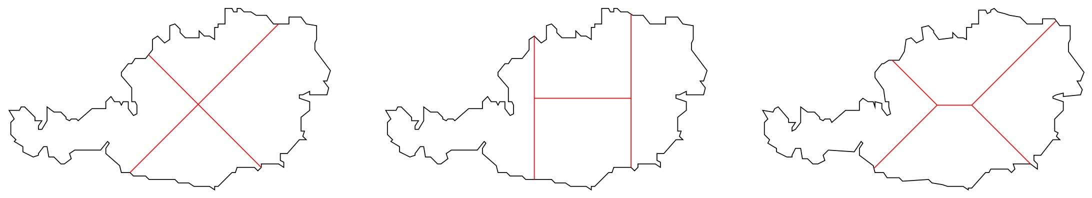


Figure 1: Three partitions of Austria by compass directions.

This paper addresses the issue of dividing a country or other geographic region into four subregions by compass directions. This is one of the aspects of the SPIRIT project, namely the determination of spatial relationships that are needed in spatial information retrieval. It will allow questions about "Northern Germany" or "Eastern Spain" to be answered properly. Note that the specified regions do not have a well-defined boundary, but are used in a loose sense. The fuzziness of geographic objects due to language issues is well-known [5, 16, 19]. Spatial relationships between two geographic objects that describe proximity or relative position are discussed in [1, 11, 17]. A partitioning into regions like the ones we compute is also useful in geographic user interfaces, where the partitioning of a country is shown and the user can select a region by clicking.

*This research is supported by the EU-IST Project No. IST-2001-35047 (SPIRIT).

Algorithmically, this paper describes how to determine — for a simple polygon P and a positive real A — all wedges with given angle and orientation that contain exactly area A of P inside. We solve both the standard version and one where the wedge is restricted to its simply-connected part inside P. The first version we solve in optimal $O(n^2)$ time and the second version in $O(n \log n)$ time.

Related in computational geometry is research on area partitionings and continuous ham-sandwich cuts of polygons [2, 3, 4, 12]. Shermer [18] shows how to partition a simple polygon by a vertical line in two equal-area halves in linear time. Díaz and O'Rourke [8, 10] show how to compute, for a simple polygon P, a point p that maximizes $\min_{h \in H} \text{Area}(h \cap P)$, where H is the set of half-planes that contain p. They also show in [9] how to partition a convex polygon into equal-size parts using an orthogonal four-partition.

In Section 2 we overview the most important criteria for a partitioning into North, East, West, and South (NEWS). This leads to three simple suggestions for good NEWS partitionings based on different criteria (shown in Figure 1). One of these is a partitioning of a simple polygon P into four equal-size parts, such that the sum of the distances of the four centers of gravity (of the NEWS regions) to the center of gravity of P is maximized. Here the distances of West and East are measured by x-coordinate only, and the distances of North and South by y-coordinate only. We show that such a partitioning has a simple shape and is unique.

In Section 3 we show that, for a given value A and simple polygon P, the set of all wedges with fixed apex angle and orientation that contain an area of size A of P has $\Theta(n^2)$ complexity, and we compute it optimally. We also develop an algorithm to compute the partitioning that maximizes the above-mentioned sum of distances of centers of gravity. For a simple polygon P with n vertices, it runs in $O(n \log n)$ time. The optimal partitioning, however, does not necessarily give connected regions for the North, East, West, and South. On the other hand, the algorithm also works for sets of polygons, like countries with islands. The efficiency is still $O(n \log n)$.

In Section 4 we discuss simply-connected partitionings into NEWS regions. We have to relax some of the criteria to obtain a problem statement that is reasonable, like the property that all four regions contain exactly 25% of the total area of P. We show that a partitioning like the right one in Figure 1 into four equal-size, simply connected regions can be computed in $O(n \log n)$ time, if one exists. This partitioning may be different from the one of Section 3. We also show that the simply-connected partitioning that minimizes the area of the largest region, or maximizes the area of the smallest region, can be computed in $O(n^3)$ time.

In Section 5 we show the output of our algorithm and compare it visually to the other suggestions for partitionings done in Section 2 using ten country outlines. Section 6 describes three possible extensions, and Section 7 gives the conclusions and open problems.

2 Criteria for a good NEWS partition

There are many criteria one can use to partition a country into four regions by compass directions. Some criteria are especially relevant for the application in query answering of geographic information retrieval, whereas others apply more to geographic user interfaces. We list these criteria next:

- the regions should be non-overlapping
- the union of the regions should fully cover P
- all regions should have the same portion of the area
- the relative orientation of any two points should be conserved (no point in North should be farther to the South than any point in South)
- the regions should be simply-connected

- the partitioning should have a simple shape
- the length of the partitioning should be small
- the partitioning should be simply-connected

Not all of these criteria can be satisfied at once. It appears to be difficult to choose criteria that would lead to 'natural' partitions for all countries. It is also unclear which criteria are conflicting or enforce each other. The fourth criterion contains the essence of the compass directions, and therefore, is necessary. The first three criteria seem important too, especially for the user interface application.

We would like to develop a simple, efficient algorithm which works well for most countries. This means we either restrict ourselves to some of the criteria, or we find a solution which meets all to a certain extent. With all of these criteria in mind, certain algorithmic problems can be formulated to find a NEWS partition of a country. They are stated in the following three suggestions. The partitionings for these suggestions are shown in Figure 1.

Suggestion 1 Compute the center of gravity of the polygon and draw two lines with slope $+1$ and -1 through this point.

Suggestion 2 Use horizontal and vertical lines to iteratively cut the polygon into four regions which each cover 25% of the polygon's area.

Suggestion 3 Divide the polygon into four equal-size regions such that the sum

$$\mathfrak{S} = dist_y(C_N, C_P) + dist_y(C_P, C_S) + dist_x(C_E, C_P) + dist_x(C_P, C_W)$$

is maximized. Here C_P denotes the center of gravity of polygon P, and C_N, C_E, C_W, C_S denote the centers of gravity of the North, East, West, and South region. $dist_x$ and $dist_y$ denote distance by x-coordinate and by y-coordinate.

There are polygons that have an optimal partitioning where the center of gravity C_P has larger y-coordinate than the center of gravity C_N of the North region. Hence, in Suggestion 3 we have to be careful how to interpret distances since they may become negative: We interpret $dist_y(C_N, C_P)$ to mean $C_{N,y} - C_{P,y}$, the difference of the y-coordinates. In the sum $\mathfrak{S}$, the coordinates of C_P disappear and we can equivalently maximize $\mathfrak{S} = dist_y(C_N, C_S) + dist_x(C_E, C_W)$.

The center of gravity can easily be computed in linear time, which solves the algorithmic part of Suggestion 1. For Suggestion 2, we let the x- and y-extent determine whether we first split by horizontal lines or first by vertical lines. We can apply the algorithm of Shermer [18] three times, which gives a linear time solution. In this paper, we will focus on finding a NEWS partitioning by an algorithm following the last suggestion. For now we will consider only the more general setting of partitioning the polygon into not necessarily simply connected NEWS regions (dealing with simply connected NEWS regions is the topic of Section 4). We will prove that a NEWS partitioning by Suggestion 3 can only lead to a partitioning with a particular shape described below.

We will denote with $\overset{\curlyvee}{\curlywedge}$ a construction made of a vertical line segment in the middle and two line segments with slope $+1$ and -1 at the top and at the bottom. Similarly, we denote with $\succ\!\prec$ the rotated shape with a horizontal line segment as middle part. We call the nodes where the three segments meet the **focal points**. In a $\succ\!\prec$ partitioning, we have a West and an East focal point. In a $\overset{\curlyvee}{\curlywedge}$ partitioning, we have a North and a South focal point.

Lemma 1 *If an arbitrary simple polygon P is divided into four (not necessarily connected) parts, such that each part covers exactly 25% of the polygon's area and the sum $\mathfrak{S} = dist_y(C_N, C_S) + dist_x(C_E, C_W)$ is maximized, then it has inner boundaries shaped $\overset{\curlyvee}{\curlywedge}$ or $\succ\!\prec$. Here C_N, C_E, C_W, C_S denote the centers of gravity of the NEWS regions. Furthermore, this partitioning is unique.*

Proof: In the proof of this lemma we make use of the following property of the center of gravity: *If a polygon P can be partitioned into two parts A and B, then C_P is the weighted average of C_A and C_B, with the areas of A and B as weights.*

We will prove in the first part that the inner boundary which separates the regions always consists of five straight line segments, one of them is vertical or horizontal and the other ones have slope $+1$ or -1. In the second part of the proof we will show that in any NEWS-division according to the lemma statement, there are at most two different focal points.

First we prove that the middle part of the inner boundary is always a horizontal (vertical) line segment. Assume that the polygon P is divided into a North N and South S region only, such that both regions cover exactly 50% of the area and the sum $\mathfrak{S}$ is optimal. Let $p_N \in int(N)$ and $p_S \in int(S)$ be two points inside N and S respectively, such that the y-coordinate of p_N is smaller than the y-coordinate of p_S. Let d denote the difference between the two y-coordinates. Let $0 < \gamma < d/2$. Because both points p_N and p_S lie in the interiors of the regions, there exist ϵ-discs $(0 < \epsilon \leq \gamma)$ around these points which are fully contained in the according region. If we exchange the two ϵ-discs, C_S moves southward and C_N moves northward, increasing the sum $\mathfrak{S}$: a contradiction. We conclude that the boundary between North and South is horizontal. With the same argument we prove that the East-West boundary is vertical.

Consider one of the other boundaries, say, the one between East E and South S. Let both regions cover exactly 50% of the area of P and assume $\mathfrak{S}' = C_{E,x} - C_{S,y}$ is maximal. Let $p_S \in int(S)$ and $p_E \in int(E)$ be two points that can be separated by a line L with slope -1, such that p_S lies above and p_E below L. There exist ϵ-discs around the points that are fully contained in the according region. If we exchange the ϵ-discs, the values $C_{E,x}$ and $C_{S,y}$. By construction, we get a total increase of $\mathfrak{S}'$. This shows that the boundary between South and East is a line with slope -1, and the proof immediately extends to the sum $\mathfrak{S}$. Because of symmetry, the NW boundary also has slope -1, whereas the NE and SW boundaries have slope $+1$. This ends the first part of the proof.

In the second part we show that always three segments of the inner boundary meet at one focal point. Because of symmetry reasons we need to look at only one case; we choose to consider how the NS, NE and SE boundaries meet. For now, we restrict ourselves to the case that each one of the NS, NE and SE boundaries intersects $int(P)$. Given an arbitrary, simple polygon P and a NEWS partition such that the three supporting lines NS, NE, SE do not meet in a single focal point. Assume that $\mathfrak{S}$ is optimal for this partitioning. For simplification we put the coordinate system such that the NS boundary lies on the x-axis, the NE boundary overlaps with the line with equation $y = x + \delta$, and the SE boundary coincides with the line with equation $y = -x + \delta$. If $\delta = 0$ then all three lines meet in one (East focal-) point (see Fig. 2 (a)). Let $\delta > 0$ (the case $\delta < 0$ is symmetric). We choose a point $p_E \in int(E)$ very close to NE, a point $p_S \in int(S)$ very close to SE and a point $p_N \in int(N)$ very close to NS. By 'very close' we mean at a distance $d = \frac{\delta}{4}$. Around these three points we draw an ϵ-disc such that $\epsilon < d$, with the same ϵ for all three points. We name these discs $D_\epsilon(p_E), D_\epsilon(p_S), D_\epsilon(p_N)$. By choosing ϵ small enough we ensure that none of these discs intersect the boundary of its region nor the lines $y = \pm x$ (see Fig. 2 (b)). We now exchange the ϵ-discs as follows: We remove $D_\epsilon(p_E)$ from E and add it to N, we remove $D_\epsilon(p_N)$ from N and add it to S and we remove $D_\epsilon(p_S)$ from S and add it to E. We claim that (a) in this new NEWS division all regions still have 25% of P's area and that (b) the new $\mathfrak{S}'$ is larger than the old $\mathfrak{S}$. (a) follows directly from our construction, and (b) is proved next. We assign coordinates to the three points as follows:

$$\begin{aligned} p_E &= (a, a + \gamma_1) \qquad && a > 0 \text{ and } 0 < \gamma_1 < \delta \\ p_S &= (b + \gamma_2, -b) \qquad && b > 0 \text{ and } 0 < \gamma_2 < \delta \\ p_N &= (-c, \gamma_3) \qquad && c > 0 \text{ and } 0 < \gamma_3 < \delta \end{aligned}$$

With this, we can compute the changes of the centers of gravity as follows:

$$\begin{aligned}\Delta C_{N,y} &= \epsilon' \cdot (a + \gamma_1 - \gamma_3)\\ \Delta C_{S,y} &= \epsilon' \cdot (-\gamma_3 - b)\\ \Delta C_{E,x} &= \epsilon' \cdot (b + \gamma_2 - a)\end{aligned}$$

with $\epsilon' > 0$, depending on ϵ and the area of P. It is easy to see that ϵ' is the same for all three equations, because N, S and E have the same area and the three (exchanged) discs have the same size.

If we sum up the three changes, we get $\Delta\mathfrak{S} = \epsilon' \cdot (\gamma_1 + \gamma_2 - 2 \cdot \gamma_3)$. By definition, γ_1, γ_2 are both > 0, and γ_3 can be chosen arbitrarily small. So, the total change $\Delta\mathfrak{S}$ is always > 0 and therefore $\mathfrak{S}' = \mathfrak{S} + \Delta\mathfrak{S} > \mathfrak{S}$. This is a contradiction to the assumption that $\mathfrak{S}$ is optimal.

Now we consider what happens if one boundary, say, NE, does not intersect $int(P)$. If NE intersects $\partial(P)$ in the East, we can use the same proof as above. If NE does not intersect $\partial(P)$ in the East, we can move NE without changing the partition. Either we get that NE, NS and SE meet at only one focal point or NE reaches the boundary of P in the East. In both cases the proof is finished.

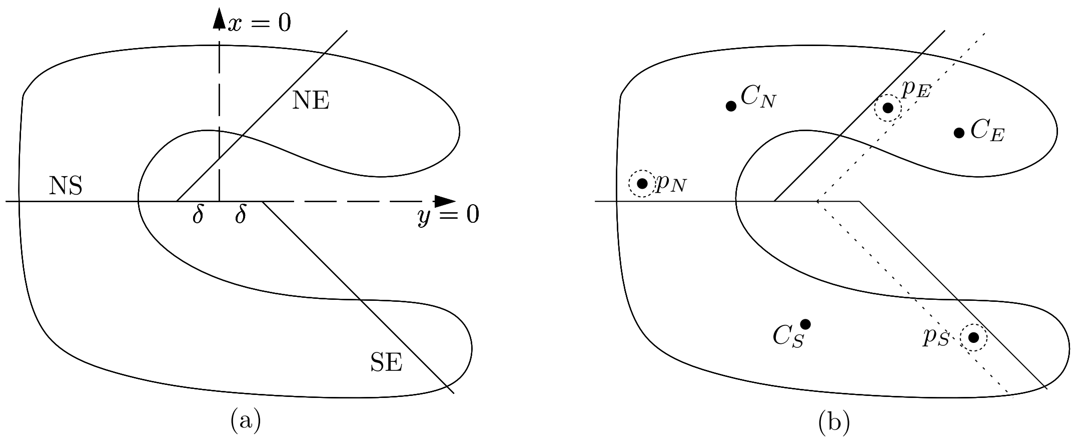


Figure 2: There are at most two focal points in any NEWS division.

To prove uniqueness, assume that two partitionings of the shape $\succ\!\prec$ exist that give four regions of area 25% of the area of P. Two different partitionings like $\succ\!\prec$ will always have one region, say, North, such that North of the one partitioning strictly contains North of the other partitioning. Since P is simply-connected, not both North regions can have exactly 25% of the area of P. Similarly, two partitionings, one of shape $\succ\!\prec$ and one of shape $\underset{\wedge}{\vee}$, will also have some region in the one partitioning for which there is proper containment in the corresponding region in the other partitioning. $\square$

3 NEWS partitions with arbitrary regions

In this section we give an algorithm to compute all wedges with fixed apex angle and orientation that contain a given area A inside. We also show how to compute a NEWS partitioning of a simple polygon following Suggestion 3.

Definition 1 *For a simple polygon P, we call a wedge of the form $(y \geq x + a) \cap (y \geq -x + b)$ that contains 25% of the area of P a* **North-wedge** *of P. East-, South-, and West-wedges are defined similarly.*

Definition 2 *The* **North-trace** *is the locus of all points that are apex of a North-wedge. East-, South-, and West-traces are defined similarly.*

The outline of an algorithm to compute a NEWS partitioning according to Suggestion 3 is as follows:

1. Compute the East-trace T_E and the West-trace T_W of polygon P.
2. Scan T_E and T_W simultaneously from top to bottom to determine if there is a pair of points $p_E \in T_E$ and $p_W \in T_W$ with the same y-coordinate and p_E to the right of p_W (or coinciding) and which gives a North area of 25% of P.
3. If such a pair exists, return the $\succ\!\!-\!\!\prec$ partitioning.
4. Otherwise, compute the North-trace T_N and the South-trace T_S.
5. Scan T_N and T_S simultaneously from left to right, to determine a pair of points $p_N \in T_N$ and $p_S \in T_S$ with the same y-coordinate and p_N higher than p_S and which gives a West area of 25% of P.
6. Return the $\curlyvee\!\!\curlywedge$ partitioning.

Before we describe the algorithm for computing a trace and finding the pair in more detail, we first show some properties of the traces. For convenience, we rotate the polygon by 45 degrees, so that a West-wedge is a quadrant of the form $(x \leq a) \cap (y \leq b)$.

Lemma 2 *After rotation, the West-trace is an infinite, continuous, piecewise quadratic curve consisting of $\Theta(n^2)$ pieces in the worst case, and any line with positive slope intersects the West-trace once.*

Proof: Let $p = (a, b)$ be a point on the West-trace, such that it is the apex of a West-wedge W that contains 25% of the area of P. Since P is a simple polygon, W must intersect the interior of P. Let $p' = (a', b')$ be any point with $a' > a$ and $b' > b$. Then a wedge with apex at p' contains strictly more than 25% of the area, thus p' cannot be on the West-trace. This proves the last statement of the lemma.

Assume next that we fix a subset of the edges of P that intersect the horizontal half-line of a West-wedge, and we fix another subset for the vertical half-line of that West-wedge. Then the curve that is this part of the trace has the form $axy + bx^2 + cy^2 + dx + ey = f$. The values of the constants a, b, c, d, e, and f depend on P.

The quadratic upper bound on the number of pieces is trivial. The lower bound construction is shown in Figure 3. Proportions are chosen such that a wedge with 25% of the area contains four of the triangular teeth. The four long diagonal lines are actually very thin parts that hardly influence the area inside the wedge. Four wedges with 25% inside are shown dashed, and the trace is shown bold. □

We next describe a sweep algorithm that computes the West-trace for the rotated polygon. Since the algorithm can be used for any fixed area of P inside the wedge, we set $A = \text{Area}(P)/4$ and give an algorithm to compute all West-wedge positions that have area A inside the intersection of P and the wedge. The sweep computes the trace from left to right.

We initialize by computing a vertical line that has area A of P left of it. Then we determine the highest point of P left of or on the vertical line. The x-coordinate of the vertical line and the y-coordinate of the highest point give the first break point p_0 of the trace. The initial part of the trace is a vertical half-line down to p_0. We initialize two lists L_x and L_y with all vertices of P sorted on increasing x-coordinate and decreasing y-coordinate, respectively. We remove the vertices from the lists up to the x- and y-coordinates of p_0.

During the sweep we maintain three pieces of information: a balanced binary search tree on the edges of P that intersect the sweep line, the leaf in this tree that stores the edge of P vertically

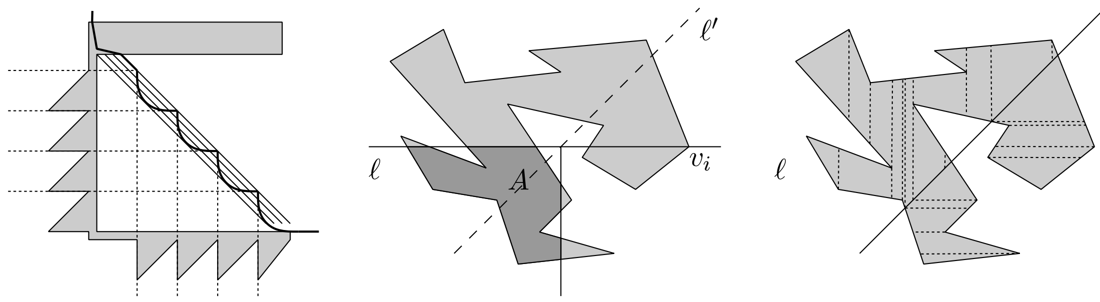


Figure 3: Left: A simple polygon that has a trace with quadratic combinatorial complexity. Middle and right: illustration of the $O(n \log n)$ time partitioning algorithm.

below the current position of the trace, and the equation of the curve that is now valid. One of three types of event can occur:

1. The vertical line through the current position of the trace (the sweep line) reaches a vertex of P.
2. The horizontal line through the current position of the trace reaches a vertex of P.
3. The trace reaches the boundary of P.

Which event will occur next can be determined in $O(1)$ time from the equation of the curve, the first element in each list, and the edges of P above and below the current trace position. Events of type 1 and 3 may require an update of the tree or the pointer to the leaf with the edge below the trace. If the event is of type 1 and the vertex of P is above the position of the trace, we do nothing else. If the event is of type 1 and the vertex of P is below the position of the trace, we output the next break point and we update the equation of the curve. For the second and third type of event similar actions are taken. Note that the sweep also continues if the position of the trace goes outside P. Later it may enter P again.

Since preprocessing takes $O(n \log n)$ time, events of types 2 and 3 take $O(1)$ time, and events of type 1 take $O(\log n)$ time, we conclude by Lemma 2:

Theorem 1 *Given a simple polygon P with n vertices and a value A, the set of all positions of apexes of wedges with a fixed shape and orientation that contain a portion of area A of P inside can be computed in optimal $O(n^2)$ time.*

The NEWS partitioning algorithm is completed by computing a pair of points on the West- and East-trace such that 25% of the area of P is in the North (step 2), or the symmetric situation (step 5). This also takes $O(n^2)$ time, so we can compute a NEWS partitioning for Suggestion 3 in $O(n^2)$ time. Note that the running time for computing a trace is $O(n \log n + k)$, where k is the complexity of the trace. Typically, k is much less than quadratic, and in practice the algorithm is efficient.

We next give a worst case $O(n \log n)$ time algorithm, which improves the previous result. However, it requires linear time triangulation [7]. We only describe the case of a $\rightarrowtail\!\!\!\prec$ partitioning. The idea is not to compute whole traces, but only a small part that has at most linear complexity. We will first determine between which two y-coordinates of vertices of P the horizontal half-line of the West-wedge lies (in the rotated polygon) for a solution that gives four regions with 25% of the area each. Similarly, we will find two consecutive x-coordinates for the vertical half-line of the West-wedge. Then we compute the West-trace only between these x- and y-coordinates.

In more detail, let v_i be a vertex of P with median y-coordinate, and consider a horizontal line ℓ through v_i (see Figure 3, middle). Compute a vertical decomposition of $P \cap \ell^-$, and determine the x-coordinate $\hat{x}$ such that $P \cap \ell^- \cap (x \leq \hat{x})$ has area A using Shermer's algorithm [18]. The West focal point $\ell \cap (x = \hat{x})$ defines a slope $+1$ line ℓ' on which the East focal point should lie as well. We intersect P with ℓ', compute a horizontal decomposition of $P \cap \ell'^-$, and a vertical decomposition of $P \cap \ell'^+$ (see Figure 3, right). Then we use a minor adaptation of Shermer's algorithm to compute the East-wedge with area A with its apex on ℓ'. This gives a $\succ\!\prec$ partitioning in case the West-wedge and the East-wedge do not overlap, which we assume first. We determine the area of the North-region. If it has area less than A, then we conclude that the desired partitioning must have a a lower y-coordinate for the apex of the West-wedge (by Lemma 2). Similarly, if the area of North is greater than A, we need a larger y-coordinate.

If the West-wedge and the East-wedge partly overlap, this is always in a square. In that case the part of P inside this square is both in West and East. To still be able to make the correct decision for the binary search, compute the area of North and of South, not including the square. North and South together have more than 50% of the area of P. If either North or South has area less than 25%, then we can still make the decision correctly. If both North and South have area more than 25%, we can conclude by the proof of Lemma 1 that no $\succ\!\prec$ partitioning gives desired result, and we compute a $\between$ partitioning instead.

We have given the basis of a binary search that will give two consecutive y-coordinates y' and y'' of vertices of P between which the West focal point must lie. Similarly, we determine between which two x-coordinates x' and x'' the West focal point must lie. Each decision step of the binary search takes $O(n)$ time, and we take $O(\log n)$ steps in total.

We now know that the West focal point of the $\succ\!\prec$ partitioning must lie in the rectangle $[x' : x''] \times [y' : y'']$. Within this rectangle we compute the West-trace explicitly; we use the sweep algorithm given before. Since no vertex of P has its x-coordinate strictly between x' and x'', nor its y-coordinate strictly between y' and y'', we only have events of type 3 in the sweep. We can have a linear number of such events if many edges of P intersect $[x' : x''] \times [y' : y'']$. However, we can not have more than $2n$ of them: the West-trace cannot intersect the same edge of P three or more times, because the subset of edges of P intersecting the half-lines of the West-wedge is the same for those three intersection points, and the function describing the trace is quadratic by Lemma 2. Hence, we compute the West-trace inside $[x' : x''] \times [y' : y'']$ in $O(n \log n)$ time altogether.

In exactly the same way we can compute the relevant part of the East-trace in $O(n \log n)$ time. By sweeping a line with slope $+1$ over these two traces we get all candidate North regions and determine if one has the desired area. If not, a $\succ\!\prec$ partitioning does not exist and we compute a $\between$ partitioning instead, in the same way.

Theorem 2 *Given a simple polygon P with n vertices, we can determine a partitioning by $\succ\!\prec$ or $\between$ where each region contains exactly 25% of the area of P in $O(n \log n)$ time. The partitioning maximizes the sum of distances of centers of gravity $\mathfrak{S} = dist_y(C_N, C_S) + dist_x(C_E, C_W)$.*

4 Simply-connected NEWS partitions

In the previous section we obtained the optimal partition into NEWS regions in the case that we are only interested in the sum of center-of-gravity distances and the property that each subregion has 25% of the total area of P. The solution of that problem may give a partition into non-connected regions. In this section we study variations where the four subregions must be simply-connected. This implies that we must relax the 25% property, allow a different shape of partitioning, or accept that a solution does not always exist.

Consider the polygon in Figure 4. It is partitioned according to the method in the previous section, giving a disconnected South. However, giving freedom to the shape of the partition only would yield the solution shown next to it, where the two former parts of South are connected artificially by a very narrow passage. Therefore, we study problems where the shape of the partitioning is maintained, but the 25% property is relaxed. However, this makes it unclear what to optimize for the sum of distances of center-of-gravity. It seems we should maximize a weighted sum, where each distance is weighted by the area of the region. At the same time we still want the

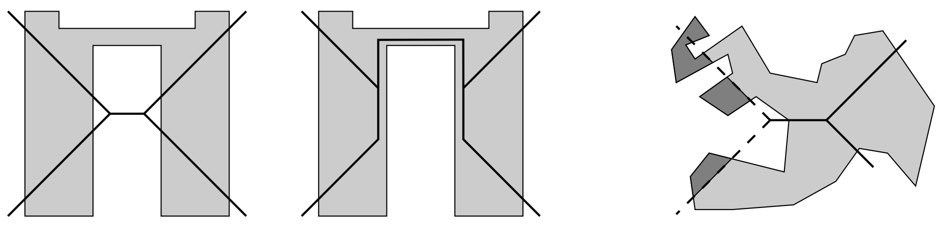


Figure 4: Example showing that it is not interesting to consider simply-connected region partitions by relaxing the shape of the partitioning only. Right: When the West focal point lies outside P, it is not well-defined how to cut. Any one of the dark regions could be West.

four NEWS regions to have area close to 25%. We choose to maintain the shape of the partition and try to obtain regions of area close to 25%. The shape of the partition will — in practice — cause the four regions to be taken from the appropriate parts of the polygon, which we previously formalized by the center-of-gravity distances. Hence, we consider the following problems:

Fair partitioning: Find a simply-connected partition into four regions by $\succ\!\!\prec$ or $\underset{\wedge}{\vee}$ such that each region has 25% of the area, or decide that no such partition exists.

Maxmin: Find a simply-connected partition by $\succ\!\!\prec$ or $\underset{\wedge}{\vee}$ that maximizes the area of the smallest region.

Minmax: Find a simply-connected partition by $\succ\!\!\prec$ or $\underset{\wedge}{\vee}$ that minimizes the area of the largest region.

We require in all cases that the partition itself be simply-connected inside P. Otherwise, we may get several options with a $\succ\!\!\prec$ or $\underset{\wedge}{\vee}$ partition. For example, if a focal point is outside, one of the diagonal boundaries incident to it cannot be used. Furthermore, a diagonal boundary, for example between North and West, may intersect the polygon several times and it seems there is a choice which cut to take (in Figure 4, right, we must choose exactly one of the four dashed intersections of $\succ\!\!\prec$ and P).

Note that for some polygons we cannot guarantee any percentage of at least $\varepsilon\%$ with $\varepsilon > 0$ fixed for all regions, no matter how small ε is. Similarly, we cannot guarantee that all regions have at most $(50 - \varepsilon)\%$ of the area. The fair partitioning problem is a special case of both the second and the third problem, but we can solve it in $O(n \log n)$ time, whereas our solution of the latter two problems takes $O(n^3)$ time.

4.1 Computing a fair partitioning

For a fair split we wish to obtain four simply-connected regions, each with 25% of the area. Again we will compute the traces for the four wedges, but we have to adapt the algorithm considerably to incorporate simply-connectedness. Interestingly, the combinatorial complexity of a trace is linear now, and we compute a trace in $O(n \log n)$ time. We again assume that P is rotated by 45 degrees and consider the computation of the West-trace. We will sweep a vertical line from left to right to compute all apex positions of wedges that have 25% of the area in the wedge. For any x-coordinate, there can be up to three such apex positions. We let $4A$ be the total area of P, so that we want to compute apex positions where the area of P in the corresponding wedge is exactly A.

Let ℓ be some position of the sweep line, and let ℓ_i be some connected component of $\ell \cap P$ (a vertical line segment). We consider apex positions on ℓ_i and observe that the area in the wedge decreases monotonically from top to bottom on ℓ_i. Assuming that no edge of P is vertical, the decrease is strict. The following four cases occur:

1. All wedges with apex on ℓ_i have area less than A.
2. All wedges with apex on ℓ_i have area greater than A.
3. There is one wedge with apex on ℓ_i that has area A.
4. None of the above, that is, the area decrease of the wedge when the apex goes from top to bottom on ℓ_i has a discontinuity where it jumps from larger than A to smaller than A.

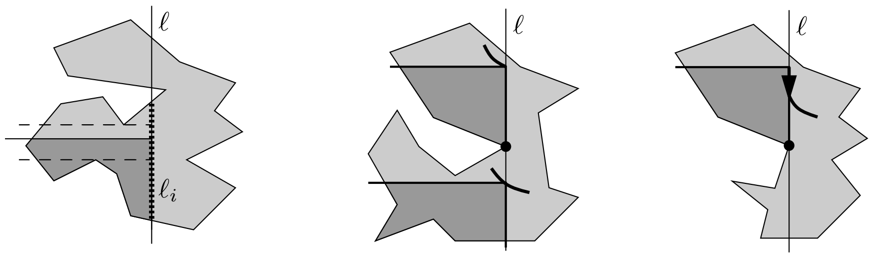


Figure 5: Left: Illustration of the area in a wedge with apex on ℓ_i (dark grey). There are discontinuities in the area decrease at the dashed lines. Middle and right: Two cases of the sweep algorithm.

The last case can occur at a y-coordinate of a vertex of P that is a local minimum or maximum of the exterior, see Figure 5, left. Corresponding to the four cases above, we store the following during the sweep:

1. We maintain the area in the wedge whose apex is at the highest point on ℓ_i, as a function of x.
2. We do not store anything.
3. We maintain the equation of the curve on which the apex of the area-A wedge lies.
4. We maintain the position on ℓ_i where the discontinuity that jumps over area A occurs. We also maintain the area of the largest wedge that has area less than A, as a function of x.

The curve of the third case is monotonically decreasing and is called the *trace*, like before. The position of the fourth case keeps its y-coordinate, that is, it follows a horizontal segment that we call *shift*.

To be able to know all events before they occur, we use a vertical decomposition *VD* and a horizontal decomposition *HD* of P. These are constructed during the preprocessing. For each edge in these decompositions, we also store the areas of the two subpolygons it induces. This part of preprocessing can easily be done in $O(n \log n)$ time (or even in linear time using linear time triangulation of polygons). For any trace or shift, we maintain in which trapezoid of *VD* and *HD* it lies at the current sweep position.

When the sweep line moves right, the following events can occur:

(i) The sweep line reaches a vertex of P.

(ii) The trace or shift reaches the boundary of P (Cases 3 and 4).

(iii) The area in the wedge of a shift becomes A (Case 4; a trace starts).

(iv) The area in the wedge with apex at the top point of ℓ_i becomes A (Case 1; a trace starts).

(v) The trace or shift crosses an edge of the vertical or horizontal decomposition (Cases 3 and 4).

Type (i) events are known in advance (some of these events are also of type (v)). Type (ii) events are known on time since we maintain in which trapezoid of *VD* and *HD* any trace or shift lies, and a trapezoid gives at most two edges of P that can be reached. Types (iii) and (iv) events are known because we maintain the function that gives the area in the wedge, and we can determine for which x an area A is reached. Type (v) events are known because we know in which trapezoid the trace or shift is.

Figure 5 shows two cases that can occur in the sweep. In the middle figure, the sweep line reaches a vertex from where the two parts of $\ell \cap P$ become connected. The trace followed by the apex of the upper area-A wedge stops, but the trace of the lower wedge continues. In the right figure, the trace followed by the apex of the area-A wedge continues after a jump vertically downward. The vertical segment of the lower part, with area less than A to the left, stops. How far the jump down is, is determined by the gain of area of the lower part, which was precomputed, and is computed by moving the apex down on ℓ and in the horizontal decomposition while maintaining the area in the corresponding wedge. At the first moment the area becomes A or less, the sweep line can continue.

To describe the full algorithm, we must describe what happens for the four cases and the five types of event in combination. Details are given in the appendix. Correctness and running time follow from the following lemmas. Lemma 4 implies linear complexity of the traces.

Lemma 3 *Every wedge with area A inside will be found by the algorithm.*

Proof: The beginning of the trace on which the apex of such a wedge lies will be found during some event. So the event handling proves the correctness. □

Lemma 4 *For any trapezoid of VD and HD, at most one trace or shift will intersect it.*

Proof: First, observe that in the vertical decomposition there cannot be two points in one trapezoid on traces with the same y-coordinate. The wedge with apex at the higher point will have strictly larger area inside. Similarly, in the horizontal decomposition, there cannot be two points in one trapezoid on traces with the same x-coordinate.

Next, consider the case when two shifts go through the same trapezoid of the vertical decomposition. Any point on a shift has the property that a point on it gives a wedge area less than A, and a point just above it gives a wedge area greater than A. Then obviously, two shifts with different y-coordinate cannot enter the same trapezoid of the vertical decomposition. It would contradict the property that the wedge area is monotonically decreasing when the apex goes down vertically. For the same reason, a shift and a trace cannot enter the same trapezoid of the vertical decomposition. □

The algorithm given above can be used for any value of A. Furthermore, it applies to any wedge with a given orientation and angle. We can simply rotate and shear-transform the polygon to bring it into the case that was described. We conclude:

Lemma 5 *Given a simple polygon P with n vertices and a value A, we can determine in $O(n \log n)$ time all locations of the apex of a wedge with fixed orientation that has area A inside it, when the intersection of the polygon and the wedge is simply-connected.*

After computing the West-traces and East-traces in the (still rotated) polygon P, we determine if there is a connecting line segment with slope $+1$ completely inside P that gives a North region of area A, and hence, a South region of area A as well. This can be done easily with a sweep over the polygon and the traces with a line of slope $+1$. If we fail, we also try the North trace, the South trace, and a connecting segment of slope -1. If this also fails, there is no partitioning of P into four simply-connected regions by ≻–≺ or ⪤, where the partitioning itself is simply-connected inside P as well.

Theorem 3 *For a simple polygon P with n vertices, we can determine in $O(n \log n)$ time if there is a simply-connected ≻–≺ or ⪤ partitioning into four simply-connected regions that all have 25% of the area of P.*

4.2 Computing a maxmin and minmax area partitioning

To compute a partitioning that maximizes the smallest area region or minimizes the largest area region, we also assume that P is rotated by 45 degrees. We only describe the case where the North and South regions are adjacent in the partitioning.

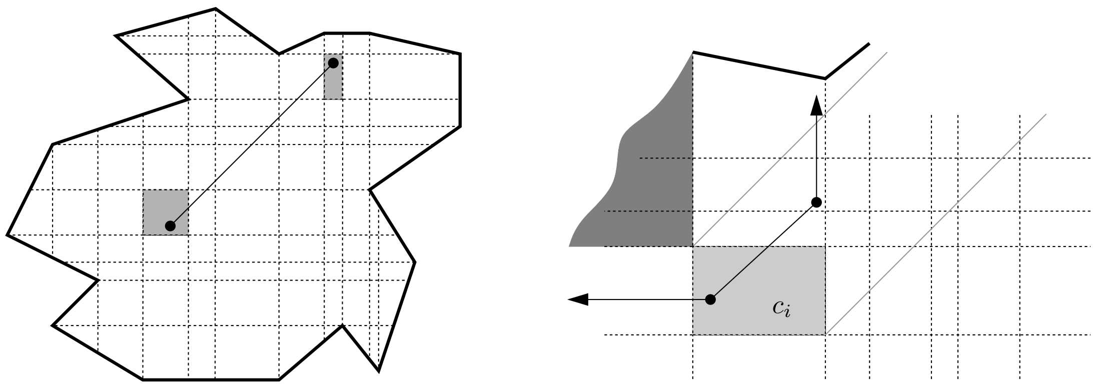


Figure 6: Left: The decomposition $\mathcal{D}$ induced by the horizontal and vertical decomposition, and a pair of aligned cells. Right: Area function for the North when the West focal point lies in c_i.

The algorithm begins with computing a horizontal decomposition and a vertical decomposition inside P. This gives a decomposition $\mathcal{D}$ of the interior of P into $O(n^2)$ cells (see Figure 6). For each vertex of the decomposition, we precompute and store the following: (i) The area of the West (East, North, and South) region if the West (East, North, and South, respectively) focal point would coincide with the vertex. (ii) The edges of P directly above, below, left, and right of it. We can easily precompute this information in $O(n)$ time per vertex, and $O(n^3)$ time overall. For every vertical edge of the decomposition $\mathcal{D}$, we precompute the area in the trapezoid left of it, and the area in the trapezoid to the right of it. This is easy in linear time per vertical edge, or in quadratic time altogether. Next, we will consider pairs of cells of the decomposition that can be aligned diagonally, by which we mean that there is a line of slope $+1$ that intersects both cells. One cell can be aligned with $\Theta(n^2)$ other cells, but we can show that there are only $O(n^3)$ alignments in total by a simple counting argument.

Lemma 6 *There are $O(n^3)$ pairs of cells in the decomposition that can be aligned diagonally.*

Proof: We imagine sweeping a line with slope $+1$ over the polygon and its decomposition. We count a pair of cells as soon as they become aligned, which is when the sweep line starts intersecting the second cell of the pair. This happens at the top left vertex of the cell (assuming the cell is in the interior of the decomposition). At any top left vertex of a cell, only $O(n)$ new pairs are found, because the sweep line intersects only $O(n)$ edges of the decomposition. $\square$

When we consider a pair of aligned cells c_i and c_j, we are interested in the functions that describe the areas of the West, East, North, and South regions as a function of the locations of the two focal points. Let these coordinates be (x_w, y_w) and (x_e, y_e), where $y_e = y_w + x_e - x_w$. The function for West is a bivariate function in x_w and y_w. The functions for East, North, and South are trivariate functions in x_w, y_w, and x_e.

Assume that we have the four area functions for a pair of cells of the decomposition. Then we determine the exact values of the three unknowns that maximize the minimum of the four functions, or minimize the maximum. It will be the highest point on the lower envelope of four 3-dimensional patches in 4-space, or the lowest point on the upper envelope. We need to solve sets of three quadratic equations in three unknowns. In the standard model of computation for surface patches, these operations take $O(1)$ time.

It remains to show that we can obtain the appropriate functions in $O(1)$ time for a pair of cells. The area functions of West and East are easy and can be obtained directly from the precomputed information. For the North and South regions, we will do this by choosing a cell for the West focal point and considering in which cells the East focal points can lie, see Figure 6. We treat the cells by looking at a row from left to right, and then going to the next higher row. In this way we can determine the appropriate functions in $O(1)$ time from the precomputed information. We obtain:

Theorem 4 *For a simple polygon P with n vertices, we can compute a simply-connected $\succ\!\!-\!\!\prec$ and $\genfrac{}{}{0pt}{}{\curlyvee}{\curlywedge}$ partitioning that minimizes the maximum area subregion or maximizes the minimum area subregion in $O(n^3)$ time.*

5 Experiments

We implemented the partitioning algorithms for all three suggestions to evaluate the performance on ten countries. We did not implement the versions producing simply-connected partitionings, so the partitionings produced for Suggestions 2 and 3 always give four regions with 25% of the area. The results can be found in Table 1.

The left column shows the partitionings for Suggestion 1, using the center of gravity. As expected, the resulting regions may have an area deviating significantly from 25%, like the East regions of Norway and Portugal and the West of Great Britain. Consequently, there are several parts that are not classified as East Portugal or West Great Britain, where intuition would do so.

The middle column shows the partitionings for Suggestion 2. If the x-extent is larger than the y-extent, the algorithm first computes a vertical line with 25% of the area left of it and a vertical line with 25% right of it. The remaining part is partitioned in equal-size regions by a horizontal line. If the y-extent is larger, the symmetric partitioning method is taken. Generally, the results correspond reasonably well to intuition. But the partitioning of a square-like shaped country as France, for instance, is highly dependent on the fact that the y-extent is slightly larger. If the x-extent were slightly larger, a completely different partitioning would be produced. As a consequence, the partitioning of France shows a West and East region that are unnaturally limited. Another unnatural situation appears in the North-East of Austria. Also observe that a large part of the West coast of Norway is in the East region.

The right column shows the partitionings for Suggestion 3. The partitionings of Austria and France are much better, but the partitionings of Norway and Italy are not ideal either.

In summary, the second and third suggestions both give good partitionings on most countries, although both fail on some. It depends on the global shape which of the two is better.

6 Extensions

In many cases it is useful to partition not only simple polygons, or to go beyond a partitioning into North, East, West, and South only. We briefly discuss three possible extensions.

Firstly, we consider non-connected polygons, which arise for countries that have one or more major islands. For all three suggestions, the algorithms can be adapted in a straightforward manner to incoporate this case. For the second and third suggestion, this follows from the fact that Shermer's algorithm [18] applies to disconnected polygons as well.

Secondly, one can define a center region of a country as well, which is important for spatial information retrieval. Suggestion 3 can be restated as partitioning a polygon into five regions, each with 20% of the area, such that the summed distances of the centers of gravity of N, E, W, and S to that of the original polygon is maximized (again, measured in x- or y-distance only). In this case, we can apply the same proof ideas as in Section 2 to show that the center region will be an axis-parallel rectangle and the boundaries between N, E, W, and S are lines with slope $+1$ or -1 that start at the corners of the central rectangle.

Thirdly, it is useful to define a degree of North, South, East, and West for any point in the country. For example, San Francisco is more in the West of the U.S. than Salt Lake City, although both can be considered to lie in the West. This is important for relevance ranking of the list of hits of the Web search.

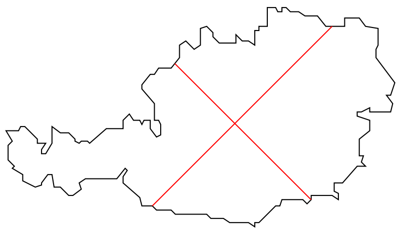
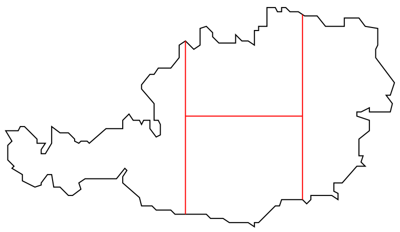
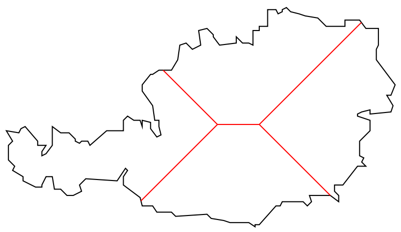
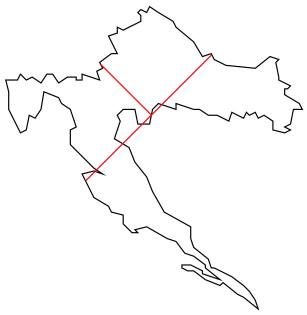
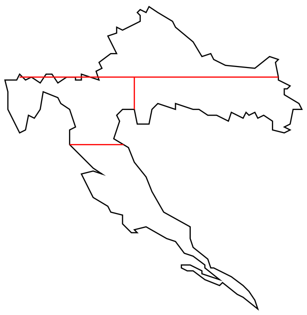
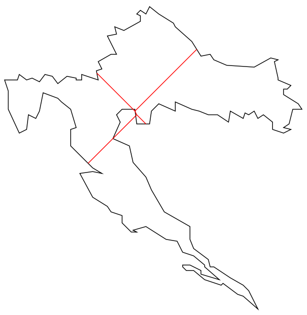
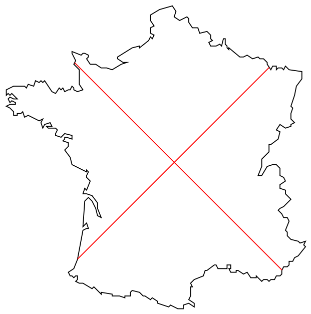
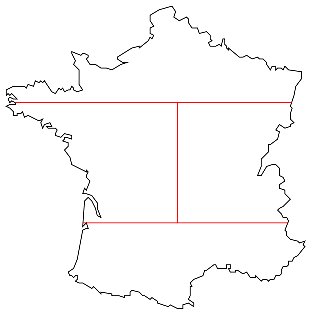
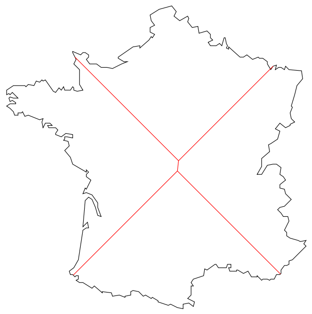
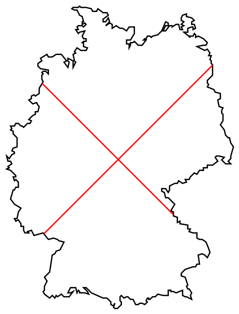
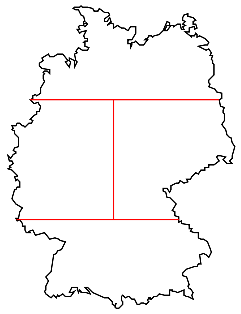
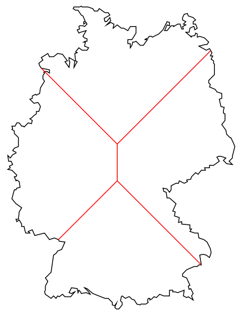
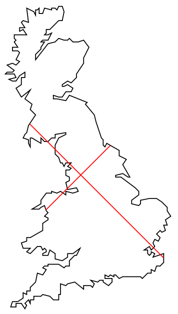
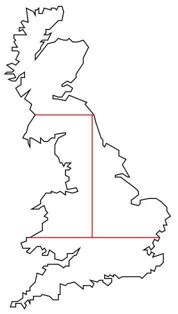
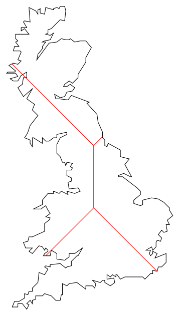
7 Conclusions and open problems

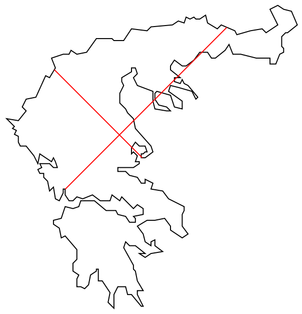
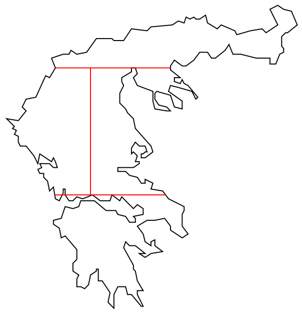
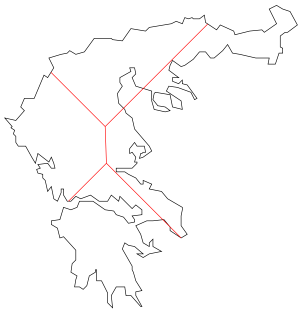
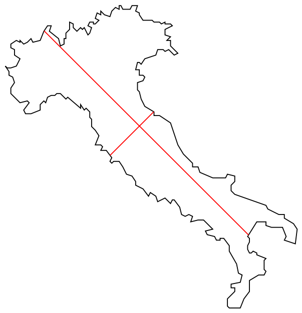
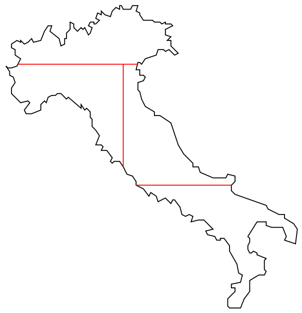
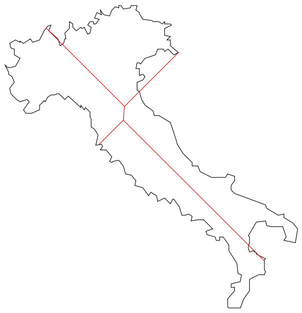
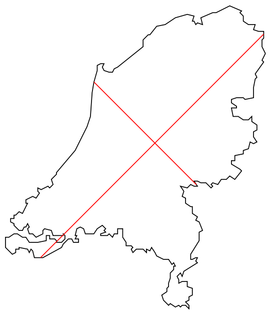
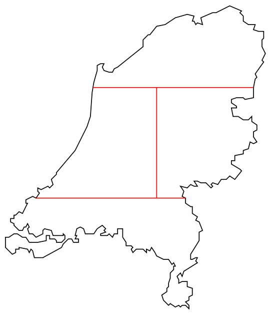
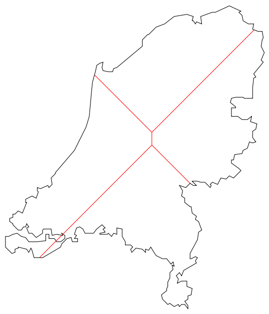
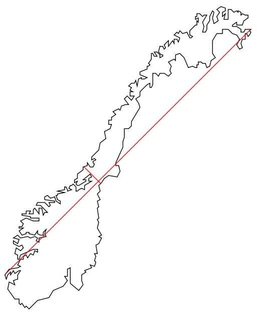
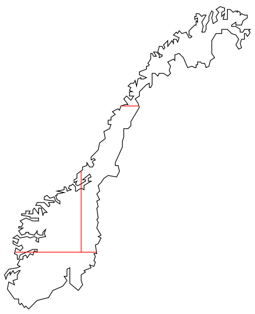
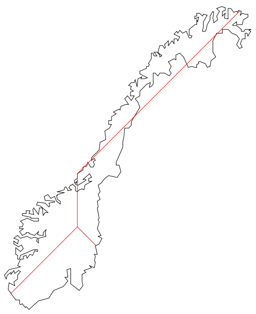
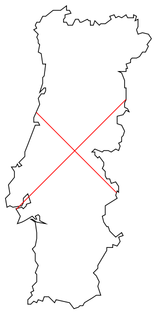
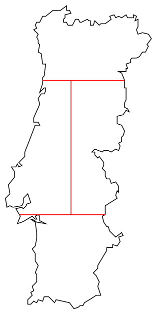
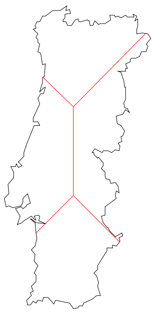
This paper discussed how to partition a region into four subregions in a natural way, such that the subregions can be associated with the North, East, West, and South. The applications lie in geographic information retrieval and Web searching. After giving criteria for a good partitioning, we suggested and analyzed three possibilities. One of these led to interesting algorithms related to earlier research on partitioning (sectioning, cutting, dividing) a simple polygon into equal-size parts. The experiments show that two of the suggestions for partitioning give good results, albeit not perfect.

Future research includes implementing the cubic time algorithm of Section 4.2, and further testing. It is open whether the algorithms in this paper can be improved: there is no $\Omega(n \log n)$ or cubic lower bound. Another interesting algorithmic question is how efficiently one can determine an orthogonal four-partition as in Suggestion 1, but one that minimizes the largest region, or maximizes the smallest region. Also the question of defining and computing a degree of North, South, East, and West is an interesting problem.

Acknowledgements The authors thank Joe O'Rourke, Danny Halperin, Frank van der Stappen, and Han-Wen Nienhuys for various comments and references.

Table 1: Comparison of three different algorithms for determining a NEWS division for ten european countries.

From top to bottom the countries are Austria, Croatia, France, Germany, and Great Britain.		
		
		
		
		
		

From top to bottom the countries are Greece, Italy, The Netherlands, Norway, and Portugal.		
		
		
		
		
		

References

[1] A.I. Abdelmoty and B.A. El-Geresy. An intersection-based formalism for representing orientation relations in a geographic database. In *2nd ACM Workshop on Advances in GIS*, pages 44–51, 1994.

[2] K.-F. Böhringer, B. Donald, and D. Halperin. The area bisectors of a polygon and force equilibria in programmable vector fields. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 457–459, 1997.

[3] P. Bose, J. Czyzowicz, E. Kranakis, D. Krizanc, and D. Lessard. Near-optimal partitioning of rectangles and prisms. In *Proc. 11th Canad. Conf. Comput. Geom.*, pages 162–165, 1999.

[4] P. Bose, J. Czyzowicz, E. Kranakis, D. Krizanc, and A. Maheswari. A note on cutting circles and squares in equal area pieces. In *Proc. FUN with Algorithms '98*, 1998.

[5] P.A. Burrough and A.U. Frank, editors. *Geographic Objects with Indeterminate Boundaries*, volume II of *GISDATA*. Taylor & Francis, London, 1996.

[6] G. Cai. GeoVSM: an integrated retrieval model for geographic information. In *Proc. 2nd GIScience*, number 2478 in Lect. Notes in Computer Science, pages 65–79, 2002.

[7] Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(5):485–524, 1991.

[8] M. Díaz and J. O'Rourke. Computing the center of area of a polygon. In *Proc. 1st Workshop Algorithms Data Struct.*, volume 382 of *Lecture Notes Comput. Sci.*, pages 171–182. Springer-Verlag, 1989.

[9] M. Díaz and J. O'Rourke. Ham-sandwich sectioning of polygons. In *Proc. 2nd Canad. Conf. Comput. Geom.*, pages 282–286, 1990.

[10] M. Díaz and J. O'Rourke. Algorithms for computing the center of area of a convex polygon. *Visual Comput.*, 10:432–442, 1994.

[11] A.U. Frank. Qualitative spatial reasoning: Cardinal directions as an example. *IJGIS*, 10(3):269–290, 1996.

[12] Susan Hert. Connected area partitioning. In *Abstracts 17th European Workshop Comput. Geom.*, pages 35–38. Freie Universität Berlin, 2001.

[13] C.B. Jones, R. Purves, A. Ruas, M. Sanderson, M. Sester, M.J. van Kreveld, and R. Weibel. Spatial information retrieval and geographical ontologies – an overview of the spirit project. In *Proc. 25th Annu. Int. Conf. on Research and Development in Information Retrieval (SIGIR 2002)*, pages 387–388, 2002.

[14] R. Larson. Geographic information retrieval and spatial browsing. In L.C. Smith and M. Gluck, editors, *Geographic Information Systems and Libraries: Patrons, Maps, and Spatial Information*, pages 81–124. 1996.

[15] M. Li, S. Zhou, and C.B. Jones. Multi-agent systems for web-based map information retrieval. In *Proc. 2nd GIScience*, number 2478 in Lect. Notes in Computer Science, pages 161–180, 2002.

[16] D.M. Mark. Toward a theoretical framework for geographic entitiy types. In A.U. Frank and I. Campari, editors, *Spatial Information Theory: A Theoretical Basis for GIS*, number 716 in Lect. Notes in Computer Science, pages 270–283. 1993.

[17] D.J. Peuquet and Z. Ci-Xiang. An algorithm to determine the directional relationship between arbitrarily-shaped polygons in the plane. *Pattern Recognition*, 20(1):65–74, 1987.

[18] T. C. Shermer. A linear algorithm for bisecting a polygon. *Inform. Process. Lett.*, 41:135–140, 1992.

[19] L. Talmy. How spoken language and signed language structure space differently. In *Proc. COSIT 2001*, number 2205 in Lect. Notes in Computer Science, pages 274–262, 2001.

[20] U. Visser, T. Vögele, and C. Schlieder. Spatio-terminological information retrieval using the BUSTER system. In *Proc. of the EnviroInfo*, pages 93–100, 2002.

Appendix: Events of the sweep for a simply-connected fair partitioning

The sweep algorithm to compute all positions of the apex so that the West-trace has 25% of the area of polygon P inside requires a case analysis for the events. Recall that we rotated P by 45 degrees, that there are four possible cases in Section 4.1 for the intersections ℓ_i of the sweep line ℓ with P, and there are five possible events. Also recall that we only consider partitionings that are simply-connected inside P, and hence, only wedges with the apex inside P are relevant. We analyze the four times five cases; note that not all combinations can occur, and there is also some overlap. The number of different types of vertex event is large, however. We begin with the less complex events.

(ii) The trace or shift reaches an edge of P. In this event, the only point or candidate point to give a desired wedge lies on the boundary of P and will go outside. Consequently, after the event, the whole interval ℓ_i will be in Case 1 (when the trace or shift leaves at the top of ℓ_i) or Case 2 (when the trace or shift leaves at the bottom of ℓ_i). We update the status structure accordingly. If we go into Case 1, we must determine the function giving the wedge area if the apex is on the top edge, and determine if and when a new trace can start (to predict event (iv) in time).

(iii) The area in the shift of a wedge becomes A. This can only happen in Case 4, and we get into Case 3 afterwards. We update the status structure and compute the equation of the trace. We also do tests to predict if events of type (v) or type (ii) will occur for the trace.

(iv) The area in the wedge with apex at the top of ℓ_i becomes A. This event can only occur for Case 1, and gives a transition into Case 3. We update the status structure and compute the equation of the trace. We also do tests to predict if and when events of type (v) or type (ii) will occur for the trace.

(v) The trace or shift crosses an edge of the vertical or horizontal decomposition. This event can only occur for Cases 3 and 4. We update the trapezoid in which the trace or shift is, and if necessary, compute the new equation of the trace or area function for the shift. When

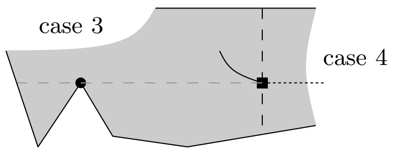


Figure 7: Type (v) event where a Case 3 to Case 4 transition occurs.

the trace crosses an edge of the horizontal decomposition, this may give a transition from Case 3 into Case 4 (see Figure 7), in which case we do the necessary updates in the status structure. To obtain the area function for the shift, we subtract the "lost area" in the subpolygon left and below the vertex that was hit. This area is precomputed as the area of some subpolygon in the horizontal decomposition.

Then we do tests do predict when the next event of types (ii), (iii), or (v) will occur for this trace or shift. All other situations to be handled are also type (i) events, and these are treated separately.

(i) The sweep line reaches a vertex of P. In several cases we have to perform a *scan down*, which is to find the y-coordinate at the current sweep line position where a trace or shift will start. We will always determine this y-coordinate from top to bottom. The *scan down* is described at the end of the appendix.

Assume that some interval ℓ_i on the sweep line ℓ reaches a vertex v of P. Then v can be at the lower endpoint of ℓ_i, at the upper endpoint of ℓ_i, or in the middle. We distinguish the following situations:

A. The edges of P incident to v are on opposite sides of the sweep line, and v is the upper endpoint of ℓ_i.

B. The edges of P incident to v are on opposite sides of the sweep line, and v is the lower endpoint of ℓ_i.

C. The edges of P incident to v are both on the left of the sweep line, and v is the lower endpoint of ℓ_i.

D. The edges of P incident to v are both on the left of the sweep line, and v is the upper endpoint of ℓ_i.

E. The edges of P incident to v are both on the right of the sweep line, and v is in the middle of ℓ_i.

These five situations must be combined with the four cases for ℓ_i before the event, as described in Section 4.1.

For situation A, the only non-trivial case is Case 1, see the top left picture in Figure 8. The area function for a wedge with the apex at the highest point on ℓ_i may grow with a jump (if both edges incident to v go upward), in which case we can stay in Case 1 or get into Case 4. We can retrieve efficiently by how much the area jumps, because we precomputed the areas of all subpolygons arising from segments in the horizontal decomposition. So this is easy to test and handle. If we get into Case 4, the shift must start at the y-coordinate of the vertex v. Situation A for Cases 2, 3, and 4 cannot give a case transition, and we only need to update a trace equation or an area function and test for future events that may occur.

For situation B, the only non-trivial case is Case 2, see the second picture in the left column of Figure 8. Depending on the edge of P to the right of v, we may get a transition into Case 4. This is easy to handle. Situation B for Cases 1, 3, and 4 cannot give a case transition.

The three other situations combined with all possible cases are shown in seven pictures in Figure 8 and in six pictures in Figure 9. In some of the pictures of Figure 9 we see that after an event we can sometimes get into Case 2 or Case 4 for the upper new intersection of ℓ and P, for instance in the pictures in the left column. Which case we get into depends on the slope of the edge of P counterclockwise from the vertex that caused the event. If the slope is negative, we get into Case 4, otherwise in Case 2 (or in the top left picture, in Case 1 or we do a scan down).

Which case we get into after any of the events in the two figures can be determined because we have precomputed all areas of subpolygons of P arising from the edges in the horizontal and vertical decomposition. For example, consider Figure 9. Let e_d be the edge vertically down from the event vertex v in the vertical decomposition, let e_u be the edge vertically up from v in the vertical

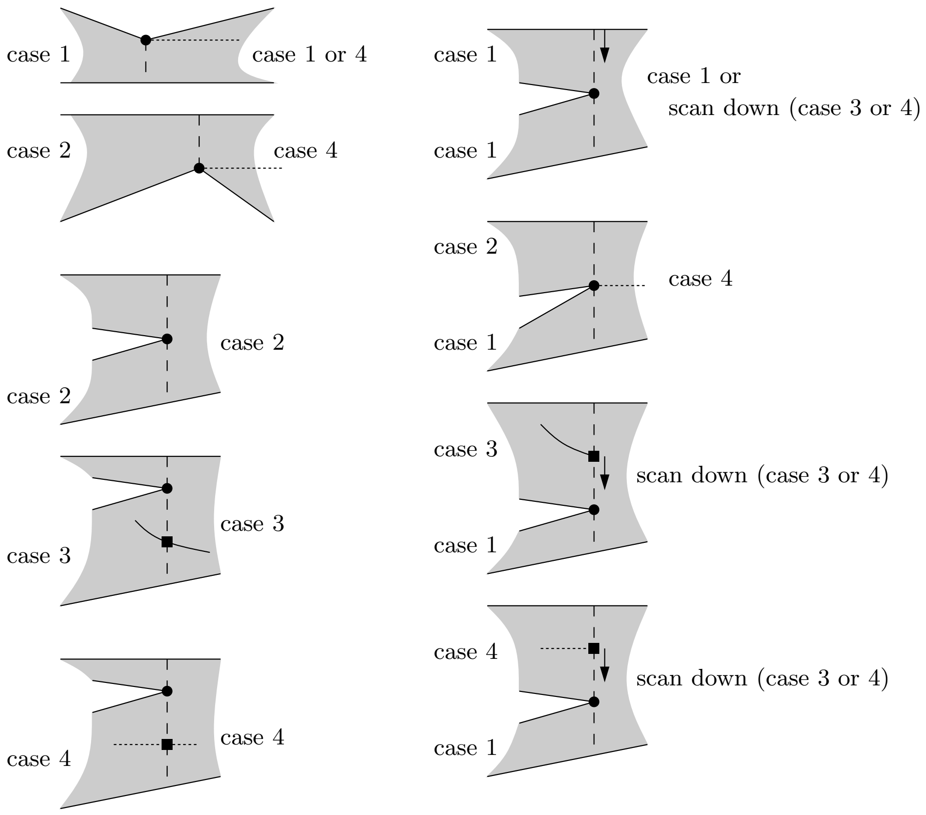


Figure 8: Vertex events in the sweep algorithm: Situations A, B, C, and D.

decomposition, and let e_l be the edge horizontally to the left in the horizontal decomposition. For each of these edges we precomputed the areas of the two subpolygons that they induce. This allows us to determine from the previous case and the subareas what case we get into for each of the two new intersections of the sweepline with P. For example, consider the bottom right picture of Figure 9 and assume the edge of P counterclockwise from v goes upward, as drawn. Then the upper branch will be in Case 2 if the area of the subpolyon below e_l is greater than A. If this area is less than A, the upper branch will be in Case 3 or 4, which is determined by a scan down starting at the end of the shift.

Scan down It remains to describe the scan down. In the scan down, we start at a point on the boundary of P or at a trace or at a shift. In all cases we can determine the area in the wedge at the start of the scan down, and this area will be greater than A. We maintain the area inside the wedge as a function of the y-coordinate of the apex.

When the apex of the wedge sweeps down, the horizontal half-line of the wedge will cross horizontal edges of the horizontal decomposition. If these are caused by a vertex w of P left of the apex, this is an event. There are three types of vertex events.

If one edge incident to w goes up and the other goes down from w, we only have to update the area function for the wedge.

If both edges incident to w go up from w, there is a sudden loss of area in the wedge. The amount is known from the areas of subpolygons in the horizontal decomposition. If the area in the wedge jumps over A, the scan down stops and a shift starts at this point. Otherwise, we update the area function and proceed with the scan down.

If both edges incident to w go down from w, there is a sudden loss of area in the wedge as well, and we do exactly the same.

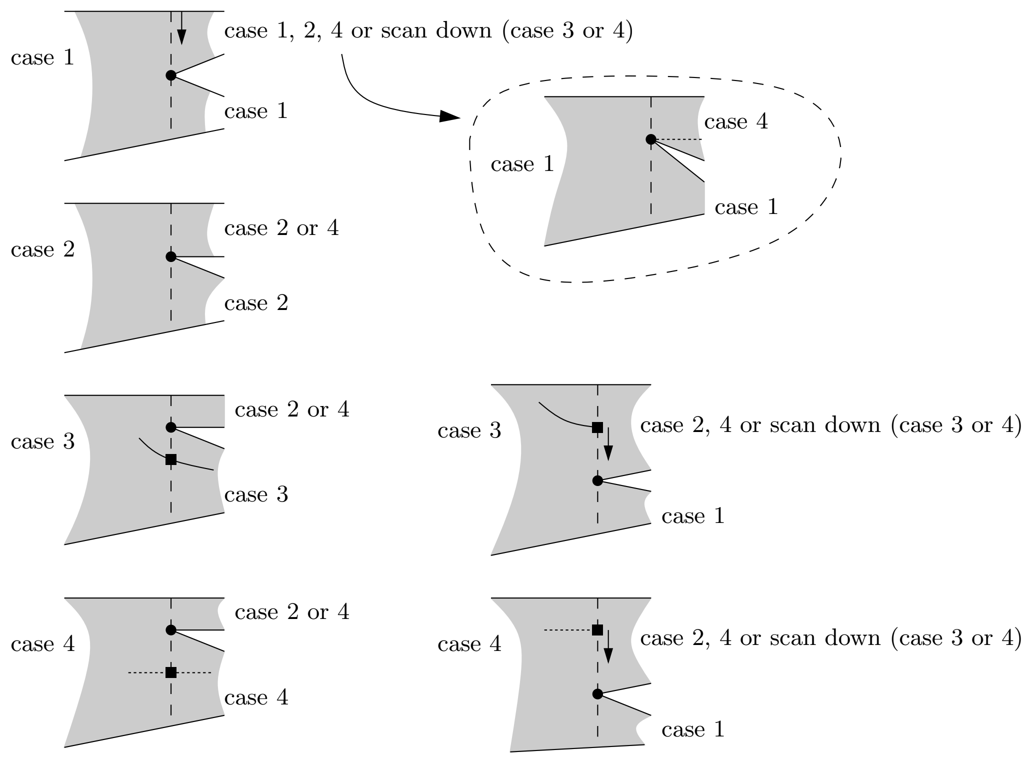


Figure 9: More vertex events in the sweep algorithm: situation E.

Besides vertex events it can also happen that the area inside the wedge reaches A. In that case, we stop the scan down and a trace starts at this point.

In the events where the area function changes, we must update the event list to detect the events where the area inside the wedge reaches A in time.